IOI 2021 Tasks（第一天试题）

目录

Contest Day 1

任务描述见后.

推荐阅读：
青少年电子信息类竞赛（1）：信息学赛事，
http://www.lynchpin.com.cn/index.php?m=content&c=index&a=lists&catid=34

第一试注意事项

对于每道题目：

- 可以在比赛系统里下载一个附件包。
- 附件包里有评测程序示例、实现示例、测试样例示例和编译脚本。
- 每道题目最多可以提交 50 次，每次只能提交一个文件。
- 在使用评测程序示例来评测你的程序时，输入数据应当符合题面所给定的格式和约束条件，否则可能会导致不确定的结果。
- 除非明确指定了其他格式，否则在评测程序示例的输入中，每一行里连续两项之间用一个空格 分隔。
- 如果你的程序没有按照题面描述的要求来做，提交至 CMS 后可能会收到“违反协议”(Protocol Violation) 的反馈信息。发生这种情况有可能是因为：
  - 从标准输入读取数据
  - 向标准输出打印内容
  - 调用 exit(0)
- 注意以上列举的行为不一定会必然导致“违反协议”的反馈信息，以及导致“违反协议”反馈信息的情况也不只有以上这些。
- 在本地机器测试你的代码时，建议使用附件包中的脚本。注意评测系统使用了 `-std=gnu++17` 编译选项。
- 如果出现无法提交到 CMS 的情况，可以使用 `ioisubmit` 工具来保存你的代码，以便在比赛结束后进行评测。
  - 在 <源代码文件> 所在的目录下运行 `ioisubmit` <题目> <源代码文件>。
  - 请你的监考人拍一张 `ioisubmit` 输出结果的照片，并发送给竞赛组织方。如果不这样做，你的提交将不会被处理。

约定

题面在给出函数接口时，会使用一般性的类型名称 `void`、`int`、`int[]`（数组）。

评测程序会采用适当的数据类型或实现，如下表所示：

语言	`void`	`int`	`int[]`	数组 `a` 的长度
C++	`void`	`int`	`std::vector<int>`	`a.size()`

限制

题目	名称	时间限制	内存限制
candies	分糖果	4.000 秒	2.00 GiB
keys	钥匙	2.000 秒	2.00 GiB
parks	喷泉公园	3.000 秒	2.00 GiB

分糖果（candies）

Khong 孔阿姨正在给附近一所学校的学生准备 n 盒糖果。盒子的编号分别为 0 到 $n-1$，开始时盒子都为空。第 i 个盒子 ($0 \leq i \leq n-1$) 至多可以容纳 $c[i]$ 块糖果（容量为 $c[i]$）。

Khong 阿姨花了 q 天时间准备糖果盒。在第 j 天 ($0 \leq j \leq q-1$)，她根据三个整数 $l[j]$、$r[j]$ 和 $v[j]$ 执行操作，其中 $0 \leq l[j] \leq r[j] \leq n-1$ 且 $v[j] \neq 0$。对于每个编号满足 $l[j] \leq k \leq r[j]$ 的盒子 k：

- 如果 $v[j] > 0$，Khong 阿姨将糖果一块接一块地放入第 k 个盒子，直到她正好放了 $v[j]$ 块糖果或者该盒子已满。也就是说，如果该盒子在这次操作之前已有 p 块糖果，那么在这次操作之后盒子将有 $\min(c[k], p + v[j])$ 块糖果。
- 如果 $v[j] < 0$，Khong 阿姨将糖果一块接一块地从第 k 个盒子取出，直到她正好从盒子中取出 $-v[j]$ 块糖果或者该盒子已空。也就是说，如果该盒子在这次操作之前已有 p 块糖果，那么在这次操作之后盒子将有 $\max(0, p + v[j])$ 块糖果。

你的任务是求出 q 天之后每个盒子中糖果的数量。

实现细节

你要实现以下函数：

```
int[] distribute_candies(int[] c, int[] l, int[] r, int[] v)
```

- c：一个长度为 n 的数组。对于 $0 \leq i \leq n-1$，$c[i]$ 表示盒子 i 的容量。
- l、r 和 v：三个长度为 q 的数组。在第 j 天，对于 $0 \leq j \leq q-1$，Khong 阿姨执行由整数 $l[j]$、$r[j]$ 和 $v[j]$ 决定的操作，如题面所述。
- 该函数应该返回一个长度为 n 的数组。用 s 表示这个数组。对于 $0 \leq i \leq n-1$，$s[i]$ 应为 q 天以后盒子 i 中的糖果数量。

例子

例 1

考虑如下调用：

```
distribute_candies([10, 15, 13], [0, 0], [2, 1], [20, -11])
```

这表示盒子 0 的容量为 10 块糖果，盒子 1 的容量为 15 块糖果，盒子 2 的容量为 13 块糖果。

在第 0 天结束时，盒子 0 有 $\min(c[0], 0 + v[0]) = 10$ 块糖果，盒子 1 有 $\min(c[1], 0 + v[0]) = 15$ 块糖果，盒子 2 有 $\min(c[2], 0 + v[0]) = 13$ 块糖果。

在第 1 天结束时，盒子 0 有 $\max(0, 10 + v[1]) = 0$ 块糖果，盒子 1 有 $\max(0, 15 + v[1]) = 4$ 块糖果。因为 $2 > r[1]$，盒子 2 中的糖果数量没有变化。每一天结束时糖果的数量总结如下：

天	盒子 0	盒子 1	盒子 2
0	10	15	13
1	0	4	13

就此情况，函数应该返回 $[0, 4, 13]$。

约束条件

- $1 \le n \le 200\,000$
- $1 \le q \le 200\,000$
- $1 \le c[i] \le 10^9$（对所有 $0 \le i \le n - 1$）
- $0 \le l[j] \le r[j] \le n - 1$（对所有 $0 \le j \le q - 1$）
- $-10^9 \le v[j] \le 10^9, v[j] \neq 0$（对所有 $0 \le j \le q - 1$）

子任务

1. （3 分） $n, q \le 2000$
2. （8 分） $v[j] > 0$（对所有 $0 \le j \le q - 1$）
3. （27 分） $c[0] = c[1] = \ldots = c[n - 1]$
4. （29 分） $l[j] = 0$ 和 $r[j] = n - 1$（对所有 $0 \le j \le q - 1$）
5. （33 分） 没有额外的约束条件。

评测程序示例

评测程序示例读入如下格式的输入：

- 第 1 行： n
- 第 2 行： $c[0]$ $c[1]$ $\ldots$ $c[n - 1]$
- 第 3 行： q
- 第 $4 + j$ 行 ($0 \le j \le q - 1$)： $l[j]$ $r[j]$ $v[j]$

评测程序示例按照以下格式打印你的答案：

- 第 1 行: $s[0]$ $s[1]$ $\ldots$ $s[n - 1]$

钥匙（keys）

建筑师 Timothy 设计了一个新的密室逃脱游戏。这个游戏里有 n 个房间，房间从 0 到 $n-1$ 编号。最开始的时候，每个房间里都恰好有一把钥匙。每把钥匙都有一个类型，钥匙的类型是一个 0 到 $n-1$ 区间内的整数。第 i 个房间里的钥匙类型是 $r[i]$。注意多个房间里可能会包含相同类型的钥匙，即 $r[i]$ 的值不一定是两两不同的。

游戏里还有 m 条**双向**的通道，通道从 0 到 $m-1$ 编号。第 j 条通道连接了一对编号不同的房间 $u[j]$ 和 $v[j]$。同一对房间之间可能存在多条通道。

参与游戏的玩家需要收集钥匙和在不同的房间之间通过通道进行移动。 当玩家使用通道 j 从房间 $u[j]$ 移动到 $v[j]$ ，或者反过来从 $v[j]$ 移动到 $u[j]$时，我们说玩家**通过**了通道 j。 只有当玩家收集到类型为 $c[j]$ 的钥匙时，玩家才可以通过通道 j。

在游戏的任意时刻，玩家可以在某个房间 x 里执行以下两种操作：

- 收集房间 x 里面的钥匙，钥匙的类型是 $r[x]$（除非对应类型的钥匙已经被收集过）。
- 通过通道 j，需要满足 $u[j] = x$ 或 $v[j] = x$，且玩家已经获得 $c[j]$ 类型的钥匙。

注意玩家收集过的钥匙可以一直使用，**永远不会被丢弃**。

最初玩家会在某个房间 s **开始**游戏，不带任何钥匙。 如果玩家从房间 s 开始，通过一系列上述描述的两种操作，能够到达房间 t，那么称房间 t 是**从房间 s 开始可以到达的**。

对于每一个房间 i ($0 \le i \le n-1$)，定义从房间 i 出发能够到达的房间数为 $p[i]$。Timothy 想要知道满足 $p[i]$ 值最小的下标 i 的集合。

实现细节

你要实现以下函数：

```
int[] find_reachable(int[] r, int[] u, int[] v, int[] c)
```

- r：一个长度为 n 的序列。对于每一个 i ($0 \le i \le n-1$)，第 i 个房间里的钥匙类型是 $r[i]$。
- u, v：两个长度为 m 的序列。 对于每一个 j ($0 \le j \le m-1$)，第 j 条通道连接了房间 $u[j]$ 和 $v[j]$。
- c：一个长度为 m 的序列。对于每一个 j ($0 \le j \le m-1$)，通过通道 j 需要用到的钥匙类型是 $c[j]$.
- 这个函数应该返回一个长度为 n 的序列 a。对于 $0 \le i \le n-1$ 中的 i，如果满足 $p[i] \le p[j] (0 \le j \le n-1)$ 那么 $a[i]$ 的值为 1，否则 $a[i]$ 的值为 0。

例子

例1

考虑以下调用：

```
find_reachable([0, 1, 1, 2],
               [0, 0, 1, 1, 3], [1, 2, 2, 3, 1], [0, 0, 1, 0, 2])
```

如果玩家从房间 0 开始游戏，可以执行以下的操作序列：

当前房间	操作
0	收集钥匙类型 0
0	通过通道 0 到房间 1
1	收集钥匙类型 1
1	通过通道 2 到房间 2
2	通过通道 2 到房间 1
1	通过通道 3 到房间 3

因此从房间 0 出发可以到达房间 3。 类似地，我们可以构造出操作序列表明所有 4 个房间都是从房间 0 出发可达的，所以 $p[0] = 4$。 下表展示了从各个房间出发可以到达的房间集合：

开始房间 i	可以到达的房间	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

所有房间中 $p[i]$ 的最小值是 2，这可以在 $i = 1$ 或 $i = 2$ 处取得。所以这次函数调用的返回值是 $[0, 1, 1, 0]$。

例2

```
find_reachable([0, 1, 1, 2, 2, 1, 2],
               [0, 0, 1, 1, 2, 3, 3, 4, 4, 5],
               [1, 2, 2, 3, 3, 4, 5, 5, 6, 6],
               [0, 0, 1, 0, 0, 1, 2, 0, 2, 1])
```

下表展示了从各个房间出发可以到达的房间集合：

开始房间 i	可以到达的房间	$p[i]$
0	$[0, 1, 2, 3, 4, 5, 6]$	7
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[3, 4, 5, 6]$	4
4	$[4, 6]$	2
5	$[3, 4, 5, 6]$	4
6	$[4, 6]$	2

所有房间中 $p[i]$ 的最小值是 2，这可以在 $i \in \{1, 2, 4, 6\}$ 处取得。所以这次函数调用的返回值是 $[0, 1, 1, 0, 1, 0, 1]$。

例3

```
find_reachable([0, 0, 0], [0], [1], [0])
```

下表展示了从各个房间出发可以到达的房间集合：

开始房间 i	可以到达的房间	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

所有房间中 $p[i]$ 的最小值是 1，这可以在 $i = 2$ 处取得。所以这次函数调用的返回值是 $[0, 0, 1]$。

约束条件

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$（对于所有的 $0 \leq i \leq n - 1$）
- $0 \leq u[j], v[j] \leq n - 1$ 且 $u[j] \neq v[j]$（对于所有的 $0 \leq j \leq m - 1$）
- $0 \leq c[j] \leq n - 1$（对于所有的 $0 \leq j \leq m - 1$）

子任务

1. （9 分）$c[j] = 0$（对于所有的 $0 \leq j \leq m - 1$）且 $n, m \leq 200$

2. （11 分） $n, m \leq 200$
3. （17 分） $n, m \leq 2000$
4. （30 分） $c[j] \leq 29$（对于所有的 $0 \leq j \leq m-1$）且 $r[i] \leq 29$（对于所有的 $0 \leq i \leq n-1$）
5. （33 分）没有额外的约束条件。

样例评分程序

评测程序示例以如下格式读取输入数据：

- 第 1 行：　n m
- 第 2 行：　$r[0]$ $r[1]$ $\ldots$ $r[n-1]$
- 第 $3+j$ 行 ($0 \leq j \leq m-1$)：　$u[j]$ $v[j]$ $c[j]$

样例评分程序按照以下格式打印 `find_reachable` 函数的返回值：

- 第 1 行：　$s[0]$ $s[1]$ $\ldots$ $s[n-1]$

喷泉公园（parks）

在附近一个公园里，有 n 座**喷泉**，编号为从 0 到 $n-1$。我们把喷泉看成是二维平面上的点。也就是说，喷泉 i（$0 \leq i \leq n-1$）是一个点 $(x[i], y[i])$，这里 $x[i]$ 和 $y[i]$ 是**偶数**。喷泉的位置各不相同。

建筑师 Timothy 受雇来规划一些**道路**的建设，以及每条道路对应的长椅的摆放。每条道路都是一个长度为 2 的**横向**或**纵向**的线段，其端点是两座不同的喷泉。游客应该能够沿着它们即可在任意两座喷泉之间互相抵达。在最开始时，公园里没有任何道路。

对于每条道路，都要在公园里摆放**恰好**一个长椅，并将其**分配给**（也就是面朝）这条道路。每个长椅必须摆放在某个点 (a, b) 上，这里 a 和 b 都是**奇数**。所有长椅的位置必须都是**不同的**。在 (a, b) 处的长椅，只能分配给两个端点均为 $(a-1, b-1)$，$(a-1, b+1)$，$(a+1, b-1)$ 和 $(a+1, b+1)$ 其中之一的道路。举例来说，在 $(3, 3)$ 处的长椅只能分配给下面四条线段所表示的道路之一：$(2, 2)$ – $(2, 4)$，$(2, 4)$ – $(4, 4)$，$(4, 4)$ – $(4, 2)$，$(4, 2)$ – $(2, 2)$。

请帮助 Timothy 判断一下，能否在满足上述所有要求的前提下，造出所有道路，并摆放和分配长椅。如果这能做到，请给他一个可行的解决方案。如果有多个满足所有要求的可行方案，你可以报告其中的任意方案。

实现细节

你要实现以下函数：

```
int construct_roads(int[] x, int[] y)
```

- x, y：长度为 n 的两个数组。对所有 i（$0 \leq i \leq n-1$），喷泉 i 是一个点 $(x[i], y[i])$，这里 $x[i]$ 和 $y[i]$ 都是偶数。
- 如果存在某个建设方案，函数应当调用 `build`（参见下文）恰好一次来报告建设方案，并紧接着返回 1。
- 否则，函数应当返回 0，并且不做 `build` 的任何调用。
- 该函数将被调用恰好一次。

你实现的函数可以调用下面的函数，以提供一个可行的道路建设与长椅摆放方案：

```
void build(int[] u, int[] v, int[] a, int[] b)
```

- 设 m 为建设方案中道路的条数。
- u, v：长度为 m 的两个数组，表示要建造的道路。这些道路的编号为从 0 到 $m-1$。对所有的 j（$0 \leq j \leq m-1$），道路 j 要连接喷泉 $u[j]$ 和 $v[j]$。每条道路必须是长度为 2 的横向或

纵向线段。任意两条不同的道路，最多只能有一个公共端点（某个喷泉）。这些道路在建成之后，必须能够沿着它们就可以在任意两个喷泉之间互相抵达。
- a, b：长度为 m 的两个数组，表示长椅。对所有的 j ($0 \leq j \leq m - 1$)，将在 $(a[j], b[j])$ 处摆放一个长椅，并且分配给道路 j。不同的长椅不能摆放在同一位置。

例子

例 1

考虑如下调用：

```
construct_roads([4, 4, 6, 4, 2], [4, 6, 4, 2, 4])
```

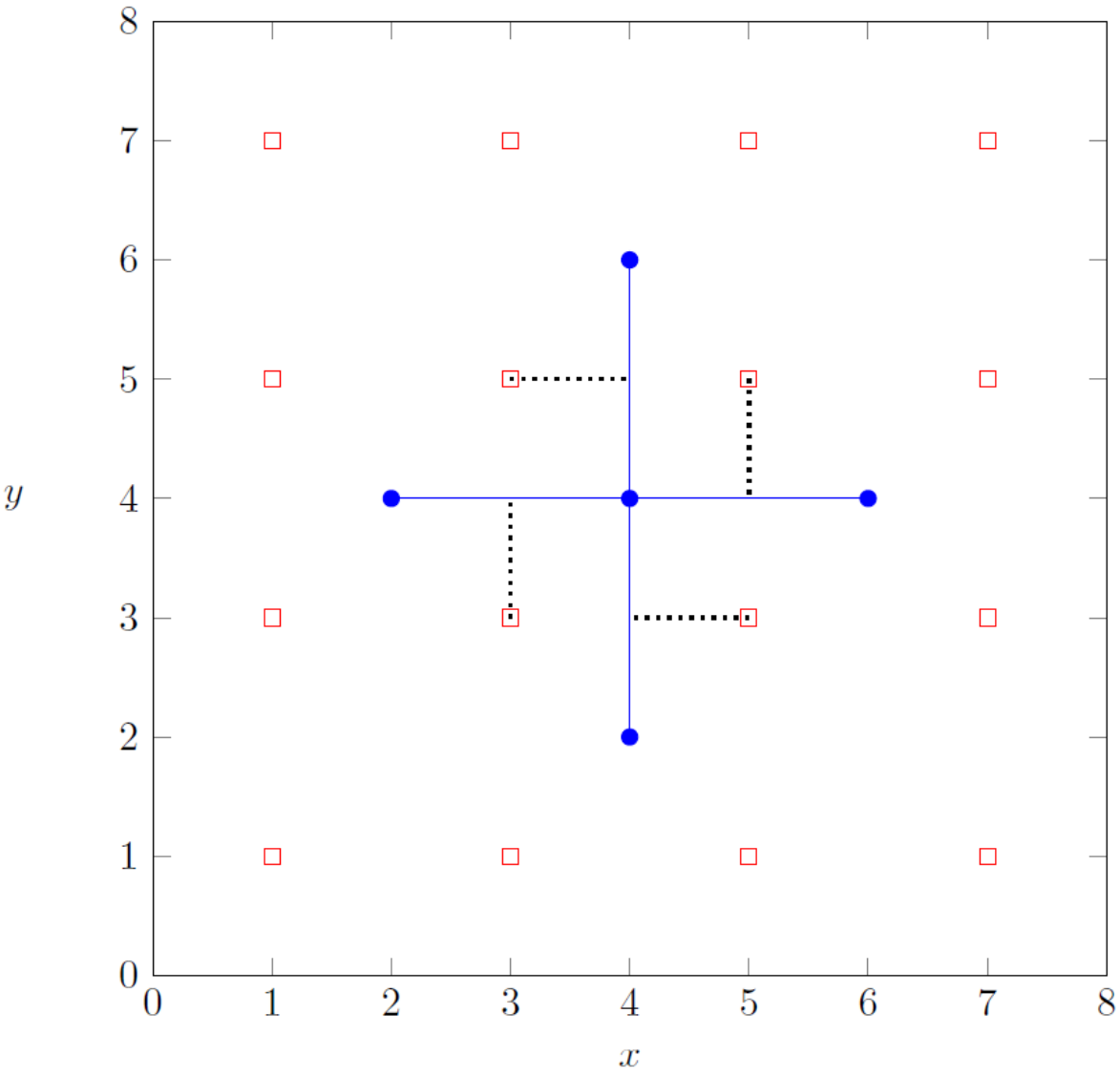
这意味着总共有 5 座喷泉：

- 喷泉 0 坐落在 $(4, 4)$ 处，
- 喷泉 1 坐落在 $(4, 6)$ 处，
- 喷泉 2 坐落在 $(6, 4)$ 处，
- 喷泉 3 坐落在 $(4, 2)$ 处，
- 喷泉 4 坐落在 $(2, 4)$ 处。

可以建造下面这样 4 条道路，其中每条道路连接两座喷泉，并且摆放着对应的长椅：

道路编号	道路所连接的喷泉的编号	所分配的长椅的位置
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

该方案对应下图：



为报告此方案，`construct_roads` 应当做如下调用：

- `build([0, 0, 3, 4], [2, 1, 0, 0], [5, 3, 5, 3], [5, 5, 3, 3])`

随后它应当返回 1。

注意，在这个例子中，有多个满足要求的方案，它们都将被视为正确。例如，调用 `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` 并返回 1，也是正确的。

例 2

考虑如下调用：

```
construct_roads([2, 4], [2, 6])
```

喷泉 0 坐落在 $(2, 2)$ 处，而喷泉 1 坐落在 $(4, 6)$ 处。由于不可能建造出满足要求的道路，`construct_roads` 应当返回 0，并且不做 `build` 的任何调用。

约束条件

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ （对于所有 $0 \leq i \leq n-1$）
- $x[i]$ 和 $y[i]$ 都是偶数（对于所有 $0 \leq i \leq n-1$）。
- 任意两座喷泉的位置均不相同。

子任务

1. （5 分） $x[i] = 2$ （对于所有 $0 \leq i \leq n-1$）
2. （10 分） $2 \leq x[i] \leq 4$ （对于所有 $0 \leq i \leq n-1$）
3. （15 分） $2 \leq x[i] \leq 6$ （对于所有 $0 \leq i \leq n-1$）
4. （20 分）至多只有一种道路建设方案，能够让游客在任意两座喷泉之间沿着这些道路即可互相抵达。
5. （20 分）任意四座喷泉都不会构成某一个 2×2 正方形的四个顶点。
6. （30 分）没有额外的约束条件。

评测程序示例

评测程序示例读取如下格式的输入：

- 第 1 行： n
- 第 $2 + i$ 行（$0 \leq i \leq n-1$）： $x[i]$ $y[i]$

评测程序示例的输出结果为如下格式：

- 第 1 行：`construct_roads` 的返回值

如果 `construct_roads` 的返回值为 1，而且调用过 `build(u, v, a, b)`，评测程序示例将额外输出：

- 第 2 行： m
- 第 $3 + j$ 行($0 \leq j \leq m-1$)： $u[j]$ $v[j]$ $a[j]$ $b[j]$